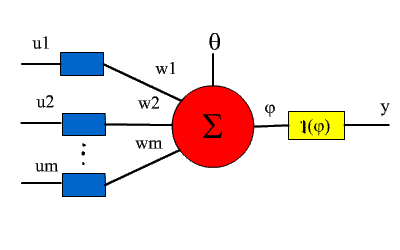
Sprawozdanie

1.Syntetyczny opis budowy oraz użytego algorytmu uczenia.

Perceptron został zaprojektowany na wzór sieci neuronowych jakie występują w naszych mózgach. Jest to matematyczne przedstawianie w jaki sposób komórki mózgowe przeliczają informacje na odpowiednie wyniki np. to że rozróżniamy kolory, albo potrafimy dodawać. Dzieje się tak za pomocą obliczania prostych funkcji i sprawdzania podanych warunków. Najpierw jest obliczana suma iloczynów wag i wejść do naszego sztucznego neuronu, a następnie wynik jest przepuszczany przez funkcję aktywacji. Najczęściej funkcjami aktywacji są proste funkcje takie jak sigma, liniowa, bipolarna, unipolarna. Nasze wejścia są to wartości dla jakich chcemy, aby nasz neutron „przetrawił” informacje, natomiast wagi są to współczynniki przez jakie przemnażamy wejścia. Następnie sprawdzany jest wynik aktywacji i w zależności od sposobu uczenia podejmowane są decyzje. Na naszych laboratoriach użyliśmy uczenia z nauczycielem czyli takiego, gdzie po każdym uczeniu jest obliczana różnica pomiędzy wynikiem otrzymanym, a prawdziwym i w zależności od przypadku zwiększamy, albo zmniejszamy wagi. Zmiana wag zależy od przyjętego algorytmu. W metodzie perceptronowej, najbardziej podstawowej, zmiana wag następuje poprzez dodanie bądź odjęcie wejść. W metodach bardziej zaawansowanych ustala się odpowiedni krok.



Gdzie:

u1, u2, um wejścia perceptronu,

w1, w2, wm wagi perceptronu,

∑ suma iloczynów wag i wejść,

θ bias,

ϕ wynik sumy,

f(ϕ) = y wynik naszej funkcji aktywacji,

Wykres przedstawiający ilość potrzebnych prób, aby perceptron wyuczył się funkcji AND (wartości posortowane)

Liczba powtórzeń uczenia 100, najmniejsza liczba prób potrzebnych to 0, która wystąpiła, aż 2 razy jak na tak krótką symulację, najwyższy wynik to 74 natomiast średnia liczba prób to 27. Uczenie to odbyło się przy współczynniku uczenia 1.

Kolejny wykres przedstawia liczbę prób potrzebnych do nauczenia dla współczynnika uczenia 1,5. Wyniki są następujące:  
Najmniej znowu okazało się, że wystarczy 0 razy powtórzyć, czyli wylosowane wagi były idealne, natomiast przy takim współczynniku uczenia się największy wynik to 68, co daje, że na 100 prób uczenia się potrzebowaliśmy, aż o 6 wyników mniej w przypadku najdłuższego uczenia się. Jednakże chociaż wykres wydaję się przedstawiać, że wyniki są mniej rozrzucone to średnio perceptron potrzebował 27,42 prób by się wyuczyć czyli o 0,42 więcej niż nasza wyjściowa wartość 1,0.

Kolejny wykres został sporządzony dla współczynnika 2,0 i osiągnął on najgorszy wynik jeżeli chodzi o średnią liczbę prób, bo aż 30,97. Zdarzyło się tak najprawdopodobniej przez zjawisko tak zwanego przeuczania się czyli, gdy perceptron zbliżał się już z odpowiednimi wagami to przekraczał optymalną wartość, a następnie musiał się cofać z matematycznego punktu widzenia i próbować znowu trafić w wartość optymalną. Dowodzi to temu, że zbyt duży czynnik uczenia się jest zły do nauczania maszynowego.

Ostatni test został przeprowadzony dla współczynnika 0,5 i chociaż ma on w 100 próbach bardzo dużo wyników poniżej 20, lub 30 prób to uzyskał on nie co mniejszą średnią liczbę prób niż współczynnik 2,0 a mianowicie 28,22.

Podsumowanie pracy.

Testy były wykonywane do momentu osiągnięcia 100% skuteczności przez dany perceptron przy wylosowanych wagach i stałych wartościach uczących. Z powodu niewystarczającej liczby danych uczących (funkcja AND posiada tylko 4 zestawy) i nie możności wyuczenia się perceptronu tylko na 3 danych uczących i sprawdzania przez 4 zdecydowałem, aby uczyć go tymi samymi 4 zestawami jak i sprawdzać go również tymi samymi 4 zestawami.

Wnioski:  
- takie proste funkcje jak AND nie są najlepszymi przykładami dla poprawnego nauczania perceptronu z powodu małej ilości danych uczących,

- proste funkcje perceptron uczy się bardzo szybko od nawet 0 liczby powtórzeń (po prostu trafiliśmy w odpowiedni zestaw wag w końcu były tylko 3 wraz z basem) do około 80 razy, ani razu w ciągu kilkuset do kilku tysięcy nie przekroczył wartości 150 powtórzeń,

- dla naszego zagadnienia najlepszy współczynnik uczenia okazał się 1, który osiągnął najmniejszy średni rezultat jak i miał najmniejszy odchył wśród wyników,

- pozostałe badane wartości czyli 2 , 1.5, oraz 0.5 nie osiągnęły znacznie gorszych rezultatów, bo tylko około 3 powtórzeń średnio potrzebowały, by się wyuczyć zadanej funkcji.

Listing kodu:

Plik Tester.h

#pragma once

class Subtester {

public:

double x1, x2, wynik;

};

class Tester {

public:

Subtester tab[4];

Tester();

};

Plik Tester.cpp

#include "Tester.h"

Tester::Tester() {

tab[0].x1 = 0;

tab[0].x2 = 0;

tab[0].wynik = 0;

tab[1].x1 = 1;

tab[1].x2 = 0;

tab[1].wynik = 0;

tab[2].x1 = 0;

tab[2].x2 = 1;

tab[2].wynik = 0;

tab[3].x1 = 1;

tab[3].x2 = 1;

tab[3].wynik = 1;

}

Plik Perceptron.h

#pragma once

#include <iostream>

#include <cstdlib>

#include <ctime>

#include "Tester.h"

class Perceptron {

private:

double w[3];

double x[2];

double wspolczynnik;

public:

Perceptron();

void ustal\_wagi(double, double, double);

void wypisz\_wagi();

void ustal\_wejscie(double, double);

double aktywacja();

void licz\_ext(double, double, double);

void licz(double, double, double);

void licz(Subtester);

void sprawdz(double,double);

bool sprawdz\_bool(double, double);

double \*losuj();

bool zrob\_test();

};

Plik Perceptron.cpp

#include "Perceptron.h";

Perceptron::Perceptron(){

w[0] = 0;

w[1] = 0;

w[2] = 0;

x[0] = 0;

x[1] = 0;

wspolczynnik = 1.0;

}

void Perceptron::ustal\_wagi(double w1, double w2, double w3) {

w[0] += w1;

w[1] += w2;

w[2] += w3;

}

void Perceptron::wypisz\_wagi() {

std::cout << "Waga 1 = " << w[0] << std::endl;

std::cout << "Waga 2 = " << w[1] << std::endl;

std::cout << "Waga 3 = " << w[2] << std::endl << std::endl;

}

void Perceptron::ustal\_wejscie(double x1, double x2) {

x[0] = x1;

x[1] = x2;

}

double Perceptron::aktywacja() {

return w[0] + w[1] \* x[0] + w[2] \* x[1];

}

void Perceptron::licz\_ext(double x1, double x2, double oczekiwany\_wynik) {

double obecny\_wynik;

ustal\_wejscie(x1, x2);

std::cout << "oczekiwany wynik: " << oczekiwany\_wynik << std::endl;

std::cout << "x1: " << x[0] << std::endl << "x2: " << x[1] << std::endl;

std::cout << "aktywacja: " << aktywacja() << std::endl;

if (aktywacja() >= 0)

obecny\_wynik = 1;

else

obecny\_wynik = 0;

std::cout << "obecny wynik:" << obecny\_wynik << std::endl;

if (obecny\_wynik > oczekiwany\_wynik) {

std::cout << "zmniejszam wagi" << std::endl;

ustal\_wagi(-1\*wspolczynnik, -x[0]\*wspolczynnik, -x[1]\*wspolczynnik);

}

else if (obecny\_wynik < oczekiwany\_wynik) {

std::cout << "zwiekszam wagi" << std::endl;

ustal\_wagi(1 \* wspolczynnik, x[0] \* wspolczynnik, x[1] \* wspolczynnik);

}

}

void Perceptron::licz(double x1, double x2, double oczekiwany\_wynik) {

double obecny\_wynik;

ustal\_wejscie(x1, x2);

if (aktywacja() >= 0)

obecny\_wynik = 1;

else

obecny\_wynik = 0;

if (obecny\_wynik > oczekiwany\_wynik)

ustal\_wagi(-1 \* wspolczynnik, -x[0] \* wspolczynnik, -x[1] \* wspolczynnik);

else if (obecny\_wynik < oczekiwany\_wynik)

ustal\_wagi(1 \* wspolczynnik, x[0] \* wspolczynnik, x[1] \* wspolczynnik);

}

void Perceptron::sprawdz(double x1, double x2) {

double tmp1 = x1, tmp2 = x2;

ustal\_wejscie(x1, x2);

if (aktywacja() >= 0)

std::cout << "TRUE" << std::endl;

else

std::cout << "FALSE" << std::endl;

ustal\_wejscie(tmp1, tmp2);

}

bool Perceptron::sprawdz\_bool(double x1, double x2) {

double tmp1 = x1, tmp2 = x2;

ustal\_wejscie(x1, x2);

if (aktywacja() >= 0) {

ustal\_wejscie(tmp1, tmp2);

return true;

}

else {

ustal\_wejscie(tmp1, tmp2);

return false;

}

}

double\* Perceptron::losuj() {

srand(time(NULL));

double \*tab = new double[3];

tab[0] = ((std::rand() % 20)-10)/10.;

tab[1] = ((std::rand() % 20)-10)/10.;

tab[2] = ((std::rand() % 20)-10)/10.;

return tab;

}

void Perceptron::licz(Subtester test) {

double obecny\_wynik;

ustal\_wejscie(test.x1, test.x2);

if (aktywacja() >= 0)

obecny\_wynik = 1;

else

obecny\_wynik = 0;

if (obecny\_wynik > test.wynik)

ustal\_wagi(-1 \* wspolczynnik, -x[0] \* wspolczynnik, -x[1] \* wspolczynnik);

else if (obecny\_wynik < test.wynik)

ustal\_wagi(1 \* wspolczynnik, x[0] \* wspolczynnik, x[1] \* wspolczynnik);

}

bool Perceptron::zrob\_test() {

if(sprawdz\_bool(1, 1));

else

return false;

if(!sprawdz\_bool(1, 0));

else

return false;

if (!sprawdz\_bool(0, 1));

else

return false;

if (!sprawdz\_bool(0, 0));

else

return false;

return true;

}

Plik Źródło.cpp

#include <iostream>

#include "perceptron.h"

using namespace std;

int main() {

Perceptron p = Perceptron();

Tester t = Tester();

double \*tab = p.losuj();

p.ustal\_wagi(tab[0], tab[1], tab[2]);

int z = 0, i = 0;

while (true) {

z = rand() % 4;

p.licz(t.tab[z]);

if (p.zrob\_test())

break;

i++;

}

cout << "Udalo sie za " << i << " podejsciem!" << endl;

p.wypisz\_wagi();

p.sprawdz(1, 1);

p.sprawdz(1, 0);

p.sprawdz(0, 1);

p.sprawdz(0, 0);

system("pause");

}

Bibliografia:

<https://pl.wikipedia.org/wiki/Funkcja_aktywacji>

<http://www.cs.put.poznan.pl/rklaus/assn/percep.htm>

<http://aragorn.pb.bialystok.pl/~gkret/SSN/Ssn_w2.PDF>

Maciej Truczka